ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.905−918
DOI: 10.1360/jos180905

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于滑动窗口的进化数据流聚类*

常建龙, 曹 锋, 周傲英+

(复旦大学 计算机科学与工程系,上海 200433)

Clustering Evolving Data Streams over Sliding Windows

CHANG Jian-Long, CAO Feng, ZHOU Ao-Ying+

(Department of Computer Science and Engineering, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-65643503, E-mail: ayzhou@fudan.edu.cn, http://www.fudan.edu.cn

Chang JL, Cao F, Zhou AY. Clustering evolving data streams over sliding windows. *Journal of Software*, 2007,18(4):905−918. http://www.jos.org.cn/1000-9825/18/905.htm

Abstract: To address the sliding window based clustering, two types of exponential histogram of cluster features, false positive and false negative, are introduced in this paper. With these structures, a clustering algorithm based on sliding windows is proposed. The algorithm can precisely obtain the distribution of recent records with limited memory, thus it can produce the clustering result over sliding windows. Furthermore, it can be extended to deal with the clustering problem over *N-n* window (an extended model of the sliding window). The evolving data streams in the experiments include KDD-CUP'99 and KDD-CUP'98 real data sets and synthetic data sets with changing Gaussian distribution. Theoretical analysis and comprehensive experimental results demonstrate that the proposed method is of high quality, little memory and fast processing rate.

Key words: evolving data stream; clustering; sliding window

摘 要: 提出了纳伪(false positive)和拒真(false negative)两种聚类特征指数直方图分别来支持纳伪误差和拒真误差窗口的聚类分析;然后,提出一种基于滑动窗口的数据流聚类方法.该方法在占用窗口大小的次线性内存空间前提下,及时保存最近数据记录的分布状况,从而实现对滑动窗口内的数据进行聚类.此外,它还可被扩展用于 *N-n* 窗口(滑动窗口的扩展模型)的数据聚类.实验采用 KDD-CUP'99 和 KDD-CUP'98 真实数据集以及变换高斯分布的人工数据集构造进化数据流.理论分析和实验结果表明,该方法具有良好的聚类质量、较小的内存开销和快速的数据处理能力.

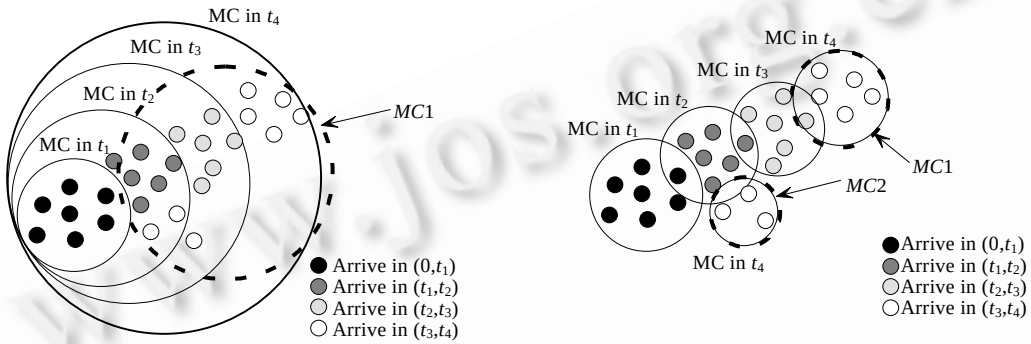
关键词: 进化数据流;聚类;滑动窗口
中图法分类号: TP311 文献标识码: A

现代计算机网络和传感器网络技术的快速发展引起了一类具有广阔发展前景的数据流应用,从金融股票交易、天气和环境监测到计算机网络监控.在该类应用中,海量流式数据以实时方式快速到达,对在线数据分析和挖掘提出了新的挑战.作为一种基础且重要的数据挖掘手段——聚类分析也越来越多地在数据流环境下被重新考虑.与传统数据聚类不同,在数据流环境下,各个簇(数据流的分布特征)往往随时间而不断进化(evolving).

* Supported by the National Natural Science Foundation of China under Grant Nos.60496325, 60496327 (国家自然科学基金)
Received 2005-12-22; Accepted 2006-06-09

现有的进化数据流聚类分析工作主要是以 CluStream 算法[1]为代表的界标窗口聚类分析算法.

在实际应用中,人们往往比较关心最近一段时间内数据流的分布状况.滑动窗口模型可被用来更好地获取当前数据流的特征.尽管 CluStream 算法可以对界标窗口内的数据进行聚类分析,然而该算法却不能适应滑动窗口下的聚类需求:首先,CluStream 中的微簇(micro-cluster)不能准确地反映当前数据流中的数据分布状况.如图 1(a)所示,在基于界标窗口的 CluStream 中,微簇的半径随界标窗口的增长而不断增大.由于没有在线淘汰“老”元组,因此只有一个微簇生成.若采用基于滑动窗口的聚类,及时地淘汰“老”元组,新到达的元组将形成两个微簇.显然,图 1(b)中的微簇相对于图 1(a)能更好地反映当前数据流中的数据分布;其次,若将 CluStream 应用到滑动窗口环境下,需要在每个新元组到达时存储一次快照(snapshot),如此巨大的处理代价和存储开销,显然难以满足数据流实时在线处理的需要.



(a) The formation of micro-clusters in CluStream
(a) CluStream 中微簇的形成过程

(b) The formation of micro-clusters in sliding winodw based clustering
(b) 基于滑动窗口的聚类中微簇的形成过程

Fig.1 The forming of mirco-clusters

图 1 微簇的形成

由于计算机和传感器的内存空间通常远小于滑动窗口的大小,在数据流应用中往往采用近似或随机算法,通过指定允许误差来换取空间上的开销(通常要求算法的空间复杂度为窗口大小的次线性).根据实际应用的需求,窗口的允许误差往往又包含两种情况:一种称为纳伪(false positive)误差,即包含有部分失效数据;另一种称为拒真(false negative)误差,即损失部分有效数据.本文的主要贡献在于:

(1) 首先,本文提出了纳伪和拒真两种聚类特征指数直方图,分别适用于纳伪误差和拒真误差的滑动窗口.聚类特征指数直方图维护滑动窗口中的聚类特征,并保证窗口内有效记录数的误差在 εN 内,其中:N 为窗口的大小;ε为用户指定的误差参数.聚类特征指数直方图的簇内结构特征可根据各簇新记录的到达情况,动态地维护指数直方图内的时间聚类特征,从而准确地捕获最近数据记录的分布状况.

(2) 其次,本文提出了一种基于滑动窗口的数据流聚类分析算法 CluWin.该算法不仅能够提供纳伪和拒真误差滑动窗口上的数据聚类结果,而且可以提供各个簇的进化过程结果.为了保证细粒度的聚类特征,CluWin 在并入新记录的同时,在线消除旧记录对簇的影响.该机制使得本文算法在聚类效果上极大地优于已有算法.

(3) 再次,CluWin 聚类算法被扩展用于解决 *N-n* 窗口内的数据流聚类问题.一些优化技术被提出用于进一步加速算法的处理速度.理论分析证明,算法的内存复杂度是窗口大小的次线性值.

(4) 最后,全面的实验证明,在滑动窗口中,CluWin 相对于现有算法,能够获取更高聚类质量,具有更小的内存空间占用量和更快的处理速度.

本文第 1 节介绍相关工作.第 2 节定义纳伪和拒真聚类特征指数直方图.第 3 节提出基于滑动窗口的聚类算法并扩展到 *N-n* 窗口的聚类.第 4 节对本文算法进行理论分析.第 5 节分析实验结果.最后是全文总结及对未来工作的展望.

1 相关工作

目前,学术界已经提出了很多数据流聚类算法.这些算法可以分为两类:单遍扫描算法[2−8]和进化分析算法[1,9−13].

单遍扫描算法把数据流聚类看作是对数据库单遍扫描的处理过程.传统的数据聚类算法[14],如 *k*-means 和 *k*-medians,被扩展到数据流环境下,并假设数据对象以数据块的方式到达[2,4,5,15].该类算法通常基于分而治之的策略,从而在小空间上获取常数因子的近似结果.例如:文献[4]采用 LOCALSEARCH 的子过程,在每个数据块到达时产生该块数据的簇中心;VFKM 算法[3]对 *k*-means 进行了扩展,并保证其产生的模型与通过无限次数据获取所产生的模型不具有太大差别;Zhou 等人提出一种用于数据流核密度估计的单遍扫描算法[8],可基于核密度估计产生聚类结果;Nam 等人提出了一种基于统计网格的方法用于单遍扫描的数据聚类[16].此外,还有 Beringer 等人研究了对并行数据流的单遍扫描聚类算法[6].

由于单遍扫描算法无法满足具有进化特征的数据流的聚类分析需求,研究者开始提出一系列进化分析算法.进化分析算法把数据流的行为看作是一个随时间不断变化的过程.Dai 等人提出了一种对多条数据流进行聚类的通用框架 COD[11].该方法可动态地对多条数据流进行聚类,并可支持多种数据挖掘的请求.Yang 考虑了一种新的多数据流聚类问题[13],在该问题中,各个数据流被看作是一个维度不断增长的向量.两条数据流间的相似性采用加权距离进行度量,并且一种增量的聚类算法被用于产生数据流的聚类结果.Aggarwal 等人提出了一种对数据流进行投影聚类的方法 HPStream[9].其主要贡献在于引入了一个衰退簇结构和对数据流进行投影聚类的思想.Zhou 等人提出一种用于跟踪滑动窗口内的簇的方法 SWClustering.有别于该项工作,本文主要讨论拒伪和纳真误差滑动窗口模型中的聚类问题,并推广到一个更普遍的 N-n 滑动窗口模型.Babcock 等人基于前人的工作[5],从理论角度对滑动窗口的聚类问题进行了研究[10].有别于该工作从理论上对聚类效果进行分析,本文主要基于滑动窗口对数据流中簇的进化过程进行挖掘.Cao 等人提出了一种基于密度的聚类算法 DenStream[17],可挖掘在有噪声环境下衰减窗口内数据流中任意形状的簇.朱蔚恒等人[18]提出一种基于空间分割的聚类方法用于挖掘具有任意形状的簇.然而,这些挖掘任意形状簇的方法并不适用于滑动窗口.

与本文最相近的工作是 Aggarwal 等人提出的 CluStream 算法[1].该算法可在界标窗口内对进化数据流进行聚类分析.然而,该算法并不适用于滑动窗口下的聚类分析问题.其主要限制在于:

(1) 该算法在滑动窗口条件下需要大量的快照(snapshot)存储开销和较大处理代价;

(2) 过期元组的影响在在线聚类过程中无法被及时消除,从而使算法的聚类效果大为降低.

2 聚类特征指数直方图

设数据流为一个不断增长的多维元组集合 $\overrightarrow{X_1}...\overrightarrow{X_j}...$,对任意 $i(i\geq 1)$, $\overrightarrow{X_i}=(x_i^1...x_i^d)$,各元组的时标为 $T_1...T_j...$,且对任意 $i<m, T_i<T_m$.滑动窗口模型中,在任意时刻只考虑并处理最近到达的 N 个元组[19].

N-*n* 窗口模型是对滑动窗口模型的扩展,它要求聚类算法可对任意的 $n(n\leq N)$个最近到达的元组进行聚类.通常,最近到达的 N 个元组称为活动元组(或有效元组),有效元组的时标称为有效时标;而其余的元组则称为过期元组,并不再参与聚类.

定义 **1**. 纳伪误差是聚类结构或聚类结果中最多包含的过期元组记录数;拒真误差是聚类结构或聚类结果中最多丢失的有效元组数.

为了对滑动窗口中的元组进行聚类,我们提出了一种近似结构,称为聚类特征指数直方图(exponential histogram of cluster features).根据直方图内引起误差的原因是包含过期元组(即纳伪误差)还是缺少有效元组(即拒真误差),又可分为纳伪聚类特征指数直方图和拒真聚类特征指数直方图.

聚类特征指数直方图根据元组的到达时标将元组划分为若干个桶(bucket).每一个桶存储该组元组的聚类特征,称为时间聚类特征.时间聚类特征除了包含聚类特征外,还含有时标信息.当聚类特征指数直方图中的第一个桶的时标不再属于当前最近的 N 个时标内时,则删除该桶.我们首先来定义纳伪时间聚类特征.

2.1 纳伪聚类特征指数直方图

定义 **2**. 一组具有时标 $T_{i_1}...T_{i_n}$ 的 d 维元组集合 $\overrightarrow{X_{i_1}}...\overrightarrow{X_{i_n}}$ 的纳伪时间聚类特征(false positive temporal cluster feature,简称 TCF)为 $2d+2$ 维的向量($\overrightarrow{CF2^x}$, $\overrightarrow{CF1^x}$,n,t).其中: $\overrightarrow{CF2^x}$ 为各元组的向量平方和,即 $\sum_{j=1}^{n}(\overrightarrow{X_{i_j}})^2$; $\overrightarrow{CF1^x}$ 为各元组的向量线性和,即 $\sum_{j=1}^{n}\overrightarrow{X_{i_j}}$;$n$ 为纳伪时间聚类特征中包含的元组个数;t 为纳伪时间聚类特征中最近元组的时标 T_{i_n}.

纳伪时间聚类特征实际上是对 Zhang 等人提出的聚类特征结构[20]作了时间上的扩展.Aggarwal 等人同样对该结构进行了扩展,但文献[1]中对时标进行累加并采用平均值来对聚类特征的时间进行估计.而本文的时间聚类特征可通过最近元组的时标值结合指数直方图结构,对聚类特征中各元组的到达时间进行限定.我们将一组元组 C 的纳伪时间聚类特征记为 $\overrightarrow{TCF}(C)$.与聚类特征结构[20]一样,纳伪时间聚类特征具有如性质 1 中所述的可加性.

性质 **1**. $\overrightarrow{TCF}(C_1\cup C_2)$ 可由 $\overrightarrow{TCF}(C_1)$ 和 $\overrightarrow{TCF}(C_2)$ 进行构建,其中,C_1 和 C_2 为两组元组集合.

证明:根据定义 2, $\overrightarrow{TCF}(C_1\cup C_2)$ 中 $\overrightarrow{CF2^x}$, $\overrightarrow{CF1^x}$ 和 n 项可以由 $\overrightarrow{TCF}(C_1)$ 和 $\overrightarrow{TCF}(C_2)$ 中的对应项直接累加获得,而 $\overrightarrow{TCF}(C_1\cup C_2)$ 中的 t 项等于 $\max\{\overrightarrow{TCF}(C_1).t,\overrightarrow{TCF}(C_2).t\}$. □

纳伪聚类特征指数直方图定义如下:

定义 **3**. 纳伪聚类特征指数直方图(exponential histogram of cluster feature,简称 EHCF)是纳伪时间聚类特征 TCF 的集合.集合中的各个纳伪时间聚类特征分别基于如下各组元组所构建:设窗口中的一组元组 $\overrightarrow{X_{i_1}}...\overrightarrow{X_{i_n}}$ 分别在 $T_{i_1}...T_{i_n}$ 到达,且当 $j<m$ 时 $T_{i_j}<T_{i_m}$.这些元组根据到达的先后次序,可划分为若干组,记作 G_1,G_2…,并满足如下约束条件:

(1) 当 $i<j$ 时,组 G_i 中的所有元组都比组 G_j 内的元组先到达.

(2) 令 $V(G)$为组 G 的大小,则 $V(G_1)=1$.对任意 $i(i>1)$,$V(G_i)=V(G_{i-1})$或 $V(G_i)=2V(G_{i-1})$.

(3) 当 $V(G)=2^i$ 时,称组 G 的级别为 i(相应地,对级别为 i 的组 G 所构建的 TCF 称为 i 级 TCF).除了最高级别的组以外,各级别均含 $\left\lceil\frac{1}{\varepsilon}\right\rceil$ 或 $\left\lceil\frac{1}{\varepsilon}\right\rceil+1$ 个组,其中,ε为用户指定的误差参数,且 $0<\varepsilon<1$.

(4) $\overrightarrow{TCF}(G_1)$ 的 t 项为当前有效时标.

当新元组 X_p 在时标 t 到达时,纳伪聚类特征指数直方图结构可采用如下步骤进行增量式维护:

(1) 根据定义2生成一个新的0级 $\overrightarrow{TCF}(C)$,其中C为仅包含X_p的数据集;

(2) 将 $\overrightarrow{TCF}(C)$ 加入到聚类特征指数直方图中.若存在 $\left\lceil\frac{1}{\varepsilon}\right\rceil$+2个0级TCF,则将最老的两个0级TCF根据性质1合并生成一个新的1级TCF.这样,0级TCF的个数就变为 $\left\lceil\frac{1}{\varepsilon}\right\rceil$.从级别$l$=1开始,若$l$级TCF的个数为 $\left\lceil\frac{1}{\varepsilon}\right\rceil$+2,则继续这种合并过程,直至某级别中TCF的个数为 $\left\lceil\frac{1}{\varepsilon}\right\rceil$ 或 $\left\lceil\frac{1}{\varepsilon}\right\rceil+1$ 时为止.

图 2 示例了 EHCF 处理以时标 1…10 到达的 X_1,…,X_{10} 这 10 个元组的维护过程.为了便于描述,ε设为 0.5(在实际应用中ε应设为远小于 1 的正数).在时标为 4 时,依据 X_4 创建了一个新的 0 级 TCF,由于 0 级 TCF 的个数为 $4\left(>\left\lceil\frac{1}{\varepsilon}\right\rceil+1=3\right)$,因而 $\overrightarrow{TCF}(\{X_1\})$ 和 $\overrightarrow{TCF}(\{X_2\})$ 合并,并生成一个新的 1 级 TCF, $\overrightarrow{TCF}(\{X_1,X_2\})$;时标为 6 和 8 时有类似的过程;时标为 10 时,X_{10} 的到达引发 $\overrightarrow{TCF}(\{X_7\})$ 和 $\overrightarrow{TCF}(\{X_8\})$ 的合并,进而引发 $\overrightarrow{TCF}(\{X_1,X_2\})$ 和 $\overrightarrow{TCF}(\{X_3,X_4\})$ 的合并.

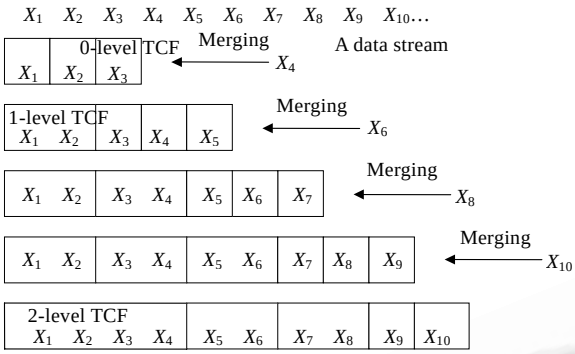


Fig.2 Merging processes of an EHCF: ε=0.5

图 2 EHCF 的并入过程:ε=0.5

性质 **2**. 纳伪聚类特征指数直方图 EHCF 的绝对误差为εN,其相对误差(即过期记录数相对总记录数的比率)为ε.

证明:由纳伪聚类特征指数直方图的定义可知,在任意时刻,纳伪聚类特征指数直方图 EHCF 的误差是由 EHCF 中最老的一个 TCF 引起的.这是因为,若最老的 TCF 的时标 t 不是当前有效时标,该 TCF 将被删除.而其他 TCF 中的元组时标必然大于 t,因而其他 TCF 所包含的记录均为有效记录.而最老的 TCF 中最多包含εN 个记录,因此,EHCF 的绝对误差为εN.其相对误差为绝对误差除以 EHCF 中包含的总记录个数,因此,EHCF 的相对误差为ε. □

定理 **1**. 设某 EHCF 由 n 个记录构成,则该 EHCF 最多包含 $\left(\frac{1}{\varepsilon}+1\right)(\log(\varepsilon n+1)+1)$ 个 TCF.

证明:设任意 EHCF 概要包含 n 个记录且误差参数为ε,我们可以根据 n 和ε构造一个指数直方图,并构造一个从 EHCF 结构到指数直方图的一一映射.根据文献[21]中的定理 2“指数直方图结构可用 $\left(\frac{1}{\varepsilon}+1\right)(\log(\varepsilon n+1)+1)$ 个桶来维护一个ε误差的概要,其中 n 为滑动窗口长度”可知,EHCF 中至多包含 $\left(\frac{1}{\varepsilon}+1\right)(\log(\varepsilon n+1)+1)$ 个 TCF. □

定理 **2**. 设某 EHCF 由 n 个记录构成,对于该 EHCF 内的任意 TCF F_i,若有 $\left\lceil\frac{1}{\varepsilon}\right\rceil(2^l-1)$个记录在 F_i 之后到达,则 F_i 可根据自身及其后的 2^l 个记录构建一个 l 级的 TCF.

证明:根据纳伪聚类特征指数直方图的定义可知,F_i 的这项操作不会破坏 EHCF 的结构,即 F_i 不违背 EHCF 定义中的 3 个约束条件.得证. □

2.2 拒真聚类特征指数直方图

拒真时间聚类特征与纳伪时间聚类特征的定义类似,主要区别在于,向量($\overrightarrow{CF2^x}$, $\overrightarrow{CF1^x}$,n,t)中的 t 项为聚类特征中最老元组的时标,其形式化定义如下:

定义 **4**. 一组具有时标 $T_{i_1}...T_{i_n}$ 的 d 维元组集合 $\overrightarrow{X_{i_1}}...\overrightarrow{X_{i_n}}$ 的拒真时间聚类特征(false negative temporal cluster feature,简称~TCF)为 $2d$+2 维的向量($\overrightarrow{CF2^x}$, $\overrightarrow{CF1^x}$,n,t).其中: $\overrightarrow{CF2^x}$ 为各元组的向量平方和,即 $\sum_{j=1}^{n}(\overrightarrow{X_{i_j}})^2$; $\overrightarrow{CF1^x}$ 为各元组的向量线性和,即 $\sum_{j=1}^{n}\overrightarrow{X_{i_j}}$;$n$ 为拒真时间聚类特征中包含的元组个数;t 为拒真时间聚类特征中最老元组的时标 T_{i_1}.

与纳伪时间聚类特征一样,拒真时间聚类特征具有性质 1 中所描述的可加性,在此不再赘述.

拒真与纳伪聚类特征指数直方图的定义类似,区别在于,拒真聚类特征指数直方图基于拒真时间聚类特征进行构建,其形式化定义如下:

定义 5. 拒真聚类特征指数直方图(简称~EHCF)是拒真时间聚类特征(~TCF)的集合.集合中的各拒真时间聚类特征分别基于如下各组元组所构建:设窗口中的一组元组 $\overrightarrow{X_{i_1}}...\overrightarrow{X_{i_n}}$ 分别在 $T_{i_1}...T_{i_n}$ 到达,且当 $j<m$ 时 $T_{i_j}<T_{i_m}$.这些元组根据到达的先后次序可划分为若干组,记作 $G_1,G_2,\ldots$,并满足如下约束条件:

(1) 当 $i<j$ 时,组 G_i 中的所有元组都比组 G_j 内的元组先到达;

(2) 令 $V(G)$为组 G 的大小,则 $V(G_1)=1$,且对任意 $i(i>1)$,$V(G_i)=V(G_{i-1})$或 $V(G_i)=2V(G_{i-1})$;

(3) 当 $V(G)=2^i$ 时,称组 G 的级别为 i.除了最高级别的组以外,各级别均含 $\left\lceil\frac{1}{\varepsilon}\right\rceil$ 或 $\left\lceil\frac{1}{\varepsilon}\right\rceil+1$ 个组,其中,ε为用户 指定的误差参数,且 $0<\varepsilon<1$;

(4) $\sim\overrightarrow{TCF}(G_1)$ 的 t 项为当前有效时标.

性质 3. 拒真聚类特征指数直方图~EHCF 的绝对误差为εN,其相对误差(即最多丢失的有效元组数相对总记录数的比率)为ε.

证明:证明过程同性质 1. □

不难看出,拒真与纳伪聚类特征指数直方图具有相似的结构和性质.由于受文章篇幅所限,我们在后面的讨论中以纳伪聚类特征指数直方图为例进行讨论.

2.3 聚类特征指数直方图的合并

一般而言,一个聚类特征指数直方图对应一个“微簇”.当系统维护许多个聚类特征指数直方图时,由于数据流中新簇的出现,往往需要对现有的两个或多个聚类特征指数直方图进行合并.根据聚类特征指数直方图所包含元组的到达次序关系,两聚类特征指数直方图 H_1 和 H_2 间的合并分为两种情况:

(1) H_1 中包含的所有元组均比 H_2 中的所有元组提前(或滞后)到达,即 H_1 与 H_2 中的元组在到达时间上不存在重叠(overlapping);

(2) H_1 中的元组与 H_2 中的元组至少有一次交错到达,即 H_1 与 H_2 中的元组在到达时间上存在重叠.

2.3.1 两个非重叠聚类特征指数直方图的合并

由于两个非重叠聚类特征指数直方图中的所有元组间存在严格的顺序关系,因而它们的合并操作相对简单.不妨设 H_1 中的所有元组均比 H_2 中的所有元组先到达,首先将 H_1 放在 H_2 的第 1 个 TCF 前,然后从 H_2 的最后一个 TCF 起依次向前扫描,并按照第 2.1 节中聚类特征指数直方图的维护操作维护一个 EHCF 结构.新生成的 EHCF 为 H_1 和 H_2 的合并结果.

2.3.2 两个重叠聚类特征指数直方图的合并

由于两个重叠聚类特征指数直方图中的元组间交织到达,无法对它们进行直接合并.因此,我们引入了一种新的结构,称为 EHCF 树,用于对两个重叠聚类特征指数直方图进行合并.

当两个重叠聚类特征指数直方图 H_1 和 H_2 开始合并时,我们并不直接对它们进行合并,而是创建一个新的聚类特征指数直方图 H_{new},且 H_{new} 维护两个指针分别指向 H_1 和 H_2.H_1 和 H_2 在形成这种结构后被称为正在被合并的 EHCF(merging-EHCF).由于 H_{new} 在创建初始时没有吸收任何元组,因而 H_{new} 不包含任何 TCF,且 H_{new} 的中心和半径被初始化为 H_1 和 H_2 中所有 TCF 累加所形成的 TCF 的中心和半径.

EHCF 树可与其他 EHCF 或 EHCF 树合并.例如,H_{new} 可能与另一个重叠的 EHCF H_3 合并,如图 3 所示.合并操作由两步构成:

(1) H_{new} 可以与 H_1 或者 H_2 直接进行合并,这是因为 H_{new} 与 H_1 或 H_2 为非重叠的,且 H_{new} 中包含的所有元组均比 H_1 或 H_2 中包含的元组晚到达.当这一步操作结束后,H_{new} 中即不再有 TCF 了.

(2) 然后,H_{new} 再与 H_3 进行合并,并生成一个新的 EHCF H'_{new} 作为 EHCF 树的新树根.

当一个新元组被 EHCF 树的树根所吸收时,EHCF 树的维护过程如下:

(1) 树根 EHCF 创建一个新 TCF 并维护其自身的 EHCF 结构.

(2) 树根的某个孩子节点创建一个 0 级的空 TCF(void-TCF,一个 EHCF 内所有的空 TCF 在实现上仅需维护一个总记数即可,因而 void-TCF 可视为不占用内存),并且好象是由它吸收了一个新的元组一样维护其自身的 EHCF 结构(称为虚拟吸收).这种虚拟吸收的过程从树根开始,在树的每一层进行,直到叶子节点结束.对于同层节点来说,虚拟吸收新元组的机会相等,除非该节点(包含它的子节点)中仅含有一个 TCF.在这种情况下,该节点虚拟吸收新元组的概率为 0,因为该节点的虚拟吸收操作不能减少整个树中的 TCF 个数.

(3) 当某节点 H_i 吸收了 $\left\lfloor \frac{1}{\varepsilon} \right\rfloor \left(2^{\lceil \log(V_{H_l}+V_{H_r}) \rceil}-1\right)$ 个元组后,其中 V_{Hl} 和 V_{Hr} 分别是其左、右孩子节点中包含的元组数,H_i 的左、右孩子节点中的 TCF 可以被合并为一个新的 TCF 并添加到 H_i 中,而 H_i 的子 EHCF 节点将被删除.

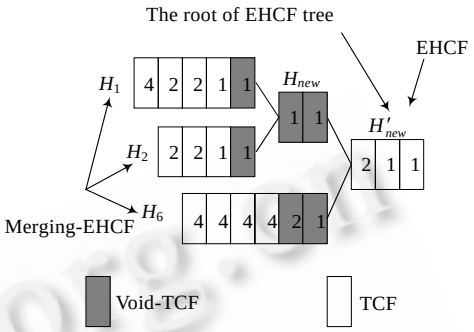


Fig.3 An EHCF-tree
图 3 一棵 EHCF 树

图 3 示例了一棵 EHCF 树,其根节点吸收了 4 个元组.根据定理 2,若某 EHCF 吸收了 $2\left\lfloor \frac{1}{\varepsilon} \right\rfloor (2^l-1)$ 个元组,则它的两个 merging-EHCF 子节点中所有 TCF 均可合并为级别大于或等于 l 的 TCF,因而,merging-EHCF 中的 TCF 个数会逐渐减少.由于 H_1 和 H_2 真正意义上的合并是在合并操作开始一段时候后才发生的,因而,我们称该过程为迟后合并.

3 基于滑动窗口的聚类

本节将讨论我们提出的 CluWin 算法,该算法可对一个滑动窗口内的数据流进行聚类分析.算法可分为两部分,如图 4 所示:第 1~18 行维护了一组 EHCF 或~EHCF 结构;第 19~22 行基于 EHCF 或~EHCF 结构生成聚类结果.CluWin 算法包含 3 个参数:*DS* 为待处理的数据流;$\varepsilon(0<\varepsilon<1)$为误差参数;*NC* 为 EHCF 的最大个数,*NC* 由系统中有效内存的大小来决定.实验证明:即使在 *NC* 为较小值时,算法仍具有较高的聚类质量.

```
Procedure CluWin(DS,ε,NC)
1: count:=0;
2: for (each record x in DS) do {
3:       Get h, the nearest EHCF/~EHCF from record x;
4:       if (x can be absorbed by h)
5:             Insert x into h;
6:       else {
7:             if (count=NC) {
8:                   Merge the two nearest EHCFs/~EHCFs;
9:                   count:=count−1;
10:            }
11:            Generate a EHCF/~EHCF containing only TCF({x});
12:            count:=count+1;
13:      }
14:      update h_e, the EHCF/~EHCF containing expired records;
15:      if (all TCFs of he are expired) {
16:            delete h_e;
17:            count:=count−1;
18:      }
19:    if (clustering request arrived) {
20:            Calculate clusters upon all of EHCFs/~EHCFs;
21:      }
22: }
```

Fig.4 The procedure of CluWin
图 4 CluWin 算法过程

3.1.1 聚类特征指数直方图组的维护

对于数据流中每一个记录 x,首先计算 x 与离 x 最近的 EHCF(记作 h)之间的距离.各 EHCF 与 x 的距离定义为各 EHCF 的中心点 c 到 x 之间的距离.为此,我们引入了如下定义:

定义 **6**. 聚类特征指数直方图 H 的中心点 c 定义为 H 中所有 $\overrightarrow{CF1^x}$ 向量的均值,即 $c=\left(\sum_{i=1}^{m}\overrightarrow{CF1^x}_i\right)\Big/\left(\sum_{i=1}^{m}n_i\right)$,其中,$H$ 包含有 m 个 TCF,分别为 $TCF_i=(\overrightarrow{CF2^x}_i, \overrightarrow{CF1^x}_i, n_i, t_i), 1\le i\le m$.

然后,算法检查 x 是否能被 h(即离 x 最近的 EHCF)所吸收.一个直观的判定方法是比较 x 和 h 的距离(记作 $dist(x,h)$)与 h 的半径 R 之间的关系,R 定义为所有被 h 所吸收记录的半径,它可以通过 h 中所维护的聚类特征值来获得.当 $dist(x,h)<\beta R$ 时(其中,β为半径的相对阈值),x 可以被 h 所吸收;反之,则说明 x 与 h 距离太远,不能被 h 所吸收.问题是,如何来确立β呢?我们知道:若数据点到中心点的距离符合正态分布,当β=2时,簇中95%以上的点将被包含在βR 边界范围以内.因而在实验中,β被设置为 2.当某个 EHCF H 仅包含一个记录时,其半径可初始化为 H 与离 H 最近的 EHCF 之间距离的 1/2.

若记录 x 可被 h 所吸收,则采用第 2.1 节中所讨论的方式将 x 并入 h.第 7~12 行说明:当 x 不能被 h 所吸收时的操作.若 EHCF 的个数未达到 NC,则可直接根据 x 构建一个新的 EHCF.若 EHCF 的个数已经达到 NC,则需要将两个最近的 EHCF 进行合并,合并过程如第 2.3 节所讨论.

第 14 行更新含有过期记录的 EHCF.由于该操作在每个新记录到达时都会被调用,因此在任何时刻,最多只有一个 EHCF(记作 h_e)含有过期元组.注意到 EHCF 是 TCF 的集合,而每个 TCF 具有一个时标,可利用 TCF 时标淘汰过期元组.即若 TCF 中的 t 值不属于最近的 N 个时标,则可删除该 TCF.若 h_e 中包含的最后一个 TCF 也被删除,则 h_e 被整体删除,且 count 减 1.可以看出,无论拒真还是纳伪时间聚类特征的淘汰策略均一致,仅仅是由于拒真和纳伪时间聚类特征中所维护的 t 的含义不同,才造成了两种不同的误差.

3.1.2 聚类结果的生成

当聚类请求到达时,CluWin 依据当前内存中的 EHCF 进行聚类(如第 19~22 行所示).对每一个 EHCF H_i,H_i 中所有的 TCF 被累加生成一个 TCF,从而获取 H_i 中包含的记录个数及 H_i 的中心.根据聚类特征向量进行聚类计算在文献[1,20]中已被广泛讨论过,其基本思想是:把任意 EHCF H_i 看作是一个落在 H_i 中心且权重为 N_i 的虚拟点,N_i 为 H_i 所包含的记录数.然后,我们可以采用加权的 k-means 算法对这些虚拟点进行聚类以获取最终结果.

图 4 是一个空间节省的算法,然而,该算法对每个新记录的处理时间开销较大.因而,我们引入如下优化技术加速算法的处理过程.在第 3 行和第 20 行中,算法需要对每个 EHCF 中的所有 TCF 进行累加.根据定理 2,某 EHCF H 中 TCF 的个数可达到 $O\left(\frac{1}{\varepsilon}\log(\varepsilon n)\right)$,其中,$n$ 为 H 所包含的记录数.因此,合并操作的时间复杂度为 $O\left(\frac{1}{\varepsilon}\log(\varepsilon n)\right)$.

为了降低这一操作的时间复杂度,我们为每个 EHCF 增量维护一个附加的 TCF 结构,从而使得合并操作的时间复杂度降低至 $O(1)$.当一个新的 TCF(记作 F')被产生并加入到 EHCF 中时(见第 5 行),F 更新为 $F+F'$.在第 14 行,EHCF 由于部分记录过期被更新时,F 同样可基于如下性质进行增量更新:

性质 **4**. 设 EHCF H 包含 m 个 TCF,$F_1,F_2,\ldots,F_m,F$ 为 H 中全部记录所构成的 TCF.当 F_1 过期时,对 H 中所有有效记录的新 TCF 概要 F'可增量生成.

证明:根据定义 2,F'中的 $\overrightarrow{CF2^x}$, $\overrightarrow{CF1^x}$ 和 n 项可以由 $\overrightarrow{TCF}(C_1)$ 和 $\overrightarrow{TCF}(C_2)$ 中的对应项直接相减获得,而 F'中的 t 项等于 F 的 t 项. □

3.1.3 N-n 窗口聚类

事实上,我们只要对 CluWin 算法略做扩展便可支持 N-n 窗口(即所有小于等于 N 长度的窗口)的聚类.用户在输入聚类请求时,需要附加输入一个表示窗口长度的参数 $n(n\le N)$.算法在根据 EHCF 概要生成聚类结果时,从每一个 EHCF 结构中找出 n 窗口范围内所有的 TCF.由于在 EHCF 结构中各 TCF 的 t 项是有序的,因而可借助

二分查找方法快速查找出各 EHCF 在 n 窗口内所有的 TCF,并根据这些 TCF 生成虚拟点,再对这些虚拟点采用加权的 k-means 算法进行聚类.

定理 **3**. 对于用户任意指定的 $n(n\le N)$窗口,CluWin 可提供以ε为相对误差的 n 窗口的聚类结果.

证明:由纳伪聚类特征指数直方图的定义可知:对于任意 $n(n\le N)$,可在任意一个 EHCF H_i 中找到一个时标上最靠近且大于 T_C-n 的 TCF 结构 F_n,从该 F_n 开始到 H_i 中最后一个 TCF 则构成了一个满足 EHCF 定义的新 EHCF H_i'.由性质 2 可知, H_i' 的相对误差为εn_i,n_i 为 H_i' 包含的记录数且 $\sum_{i=1}^{k}n_i=n$.由于各 H_i' 的绝对误差为εn_i,基于纳伪聚类特征指数直方图的 CluWin 算法的相对误差为 $\left(\sum_{i=1}^{k}\varepsilon n_i\right)\Big/n=\left(\sum_{i=1}^{k}\varepsilon n_i\right)\Big/\sum_{i=1}^{k}n_i=\varepsilon$.同理可证,基于拒真聚类特征指数直方图的 CluWin 算法可提供以ε为相对误差的 n 窗口的聚类结果. □

4 理论证明

设 H_{new} 是在 H_1 和 H_2 进行迟后合并时生成的新 EHCF,且 $V_{H_1}=m_1,V_{H_2}=m_2,V_{H_{new}}=n$,其中,n 随着 H_{new} 不断吸收新记录而不断增加.H_1,H_2 和 H_{new} 中所包含的 TCF 总个数为 B_{com}.设 H_0 为一个包含 m_1+m_2+n 个记录的 EHCF,根据定理 1,H_0 中包含的 TCF 个数为 $B_{H_0}=\left(\frac{1}{\varepsilon}+1\right)(\log(\varepsilon(m_1+m_2+n))+1)$.根据定理 1,随着 n 的增大,B_{com} 和 B_{H_0} 之间的差异会越来越小.根据定理 2,两个 merging-EHCF 可以合并到 H_{new} 中,这时,在迟后合并的 EHCF 和 H_0 之间已没有差异,它们将占用相同大小的内存.由于迟后合并所引起的内存空间差异会逐渐消失,下面的分析不考虑这一影响.定理 1 说明,EHCF 是一个次线性空间的结构.我们可以获得如下定理,证明 CluWin 算法的空间复杂度是窗口大小的次线性值.

定理 **4**. CluWin 算法最多使用 $O\left(\frac{k}{\varepsilon}\log\left(\varepsilon\left\lceil\frac{N}{k}\right\rceil\right)\right)$ 个 TCF,其中:N_i 表示 H_i 中所包含的记录数,且 $\sum_{i=1}^{k}N_i=N$; N 为窗口长度;k 为 EHCF 的个数.

证明:根据定理 1,每一个 EHCF H_i 最多包含 $\left(\frac{1}{\varepsilon}+1\right)(\log(\varepsilon N_i+1)+1)$ 个 TCF.因此,整个算法在内存中维护的 TCF 个数至多为 $\sum_{i=1}^{k}\left(\frac{1}{\varepsilon}+1\right)(\log(\varepsilon N_i+1)+1)=\left(\frac{1}{\varepsilon}+1\right)\left(\log\left(\prod_{i=1}^{k}(\varepsilon N_i+1)+k\right)\right)$.若 k 个自然数 $N_1,N_2,\ldots,N_k$ 满足 $\sum_{i=1}^{k}N_i=N$,则 $\prod_{i=1}^{k}N_i\le\left\lceil\frac{N}{k}\right\rceil^k$,因而 $O\left(\frac{1}{\varepsilon}\log\left(\prod_{i=1}^{k}\varepsilon N_i\right)\right)=O\left(\frac{1}{\varepsilon}\log\left(\varepsilon^k\left\lceil\frac{N}{k}\right\rceil^k\right)\right)=O\left(\frac{k}{\varepsilon}\log\left(\varepsilon\left\lceil\frac{N}{k}\right\rceil\right)\right)$.可得 CluWin 算法最多使用 $O\left(\frac{k}{\varepsilon}\log\left(\varepsilon\left\lceil\frac{N}{k}\right\rceil\right)\right)$ 个 TCF. □

由于 CluStream 需要在金字塔时间存储当前内存中的微簇快照,令快照中微簇个数为 Mk,M 通常为大于或等于 10 的整数,根据金字塔时间框架的定义,当到达 N 条记录时,算法需要存储 $O\left(\frac{1}{\varepsilon}\log(N)\right)$ 个快照.因而,若以微簇为最小内存单元,算法的空间复杂度为 $O\left(\frac{Mk}{\varepsilon}\log(N)\right)$.由于 M 通常大于或等于 10,因而 CluWin 算法的空间复杂度比 CluStream 少一个数量级.

此外,由于 CluStream 算法需要在金字塔时间到达时对在内存中的所有微簇存储一个快照,因此,该存储操作是一种非常耗时的操作,时间复杂度为 $O(Mk)$;在 CluWin 算法中,则只需要对一个 EHCF(或~EHCF)结构进行调整即可,该操作的时间代价相对较少.如前所述,我们为每个 EHCF(或~EHCF)增量维护一个附加的 TCF 结构,从而使得调整操作的时间复杂度为 $O(1)$.因而 CluWin 相对 CluStream 具有更低的时间复杂度.

5 实验分析

5.1 实验设置

所有实验在一台 PentiumIV 2.4GHz 的 PC 上进行,操作系统为 Windows 2000 专业版.为了证明 CluWin 算法的有效性,我们采用著名的 CluStream 算法作为比较算法.算法用 Visual C++实现.

实验数据包括真实数据集和人工数据集.由于人工数据集在生成时可以控制簇的个数、数据点的维数、数据点个数以及各种分布或进化特性,我们采用人工数据集进行算法内存开销和处理时间测试.

为了测试 CluWin 的聚类效果,真实数据集采用 KDD-CUP'99 网络入侵检测数据集,该数据集曾被多篇数据流聚类文章所引用,例如:Aggrwal 等人[1]利用该数据集比较 CluStream 和 STREAM 算法;Chalaghan 等人[4]用它来比较 STREAM 和 Birch 算法.该数据集由一个局域网中 TCP(transfer control protocol)连接的原始记录所构成.各记录含有连接的持续时间、从源到目的传输的字节数等.数据集中每条记录对应于一个正常的连接或者是 4 种类型的网络攻击之一,例如,拒绝服务攻击、非授权访问远程机器攻击等.与文献[1,4]一样,34 个连续属性从 42 个有效属性中被取出,用于数据聚类.另一个相对稳定的真实数据集是 KDD-CUP'98,该数据集记录关于慈善捐赠的信息,共 95 412 条.对该数据集进行聚类分析,可用于揭示具有相似捐赠行为的捐助群体.与文献[1]一样,我们从 481 个字段中抽取 56 个字段,并采用记录的输入顺序作为数据流的记录顺序.

人工数据集满足一系列高斯分布.与文献[1]中的方法类似,每 10K 个数据点改变一次当前高斯分布的中心和方差,且采用如下标记描述人工数据集:'*B*'表示数据集中包含的数据点数;'*C*'表示簇的个数;'*D*'表示数据点的维数.例如,*B*100*C*20*D*30 表示数据集包含 100K 个数据点,分别属于 20 个不同的簇,各数据点的维数为 30.

聚类质量采用聚类分析中最常用的平方距离和(sum of squared distance,简称 SSQ)作为度量指标,具体定义如下[1]:设当前时标为 T_c,窗口大小为 N,对各个处于 $T_c–N$ 到 T_c 间的数据点 p_i,首先找到与它最近的中值点 C_{pi};然后计算 p_i 和 C_{pi} 之间的距离 $dist(p_i,C_{pi})$;最后计算在时刻 T_c 窗口大小为 N 的 SSQ,即对所有处于 $T_c–N$ 到 T_c 间的数据点 p_i 累加 $dist^2(p_i,C_{pi})$.

在实验中,CluWin 中的 EHCF 最大个数与 CluStream 中的最大微簇个数相等.对于 CluStream 采用与文献[1]中相同的设置.若微簇的相对时标不处于最近的 N 个时标范围内,则删除该微簇.除非特别指明,CluWin 算法中的参数设置如下:误差参数 ε=0.1,窗口大小 N=10000,半径相对阈值 β=2.实验表明,纳伪误差窗口聚类与拒真误差窗口聚类在内存消耗、处理时间和参数敏感性上具有几乎相同的特性.因此,我们在分析这些指标时以纳伪误差窗口聚类为例.

5.2 聚类效果

首先,我们在 KDD-CUP'99 网络入侵检测数据集的不同时刻和不同大小窗口中比较 CluWin 和 CluStream 的聚类效果.为了使结果更准确,算法各执行 5 次并计算平均 SSQ 作为结果值.图 5 为窗口大小为 10 000 时的聚类结果.注意到纵轴为指数增长的刻度,CluWin 比 CluStream 有几个数量级的性能提高.例如,在时标 200 000 处,CluWin 的 SSQ 要 10^5 倍地小于 CluStream.图 6 显示 CluWin 在 KDD-CUP'98 这个相对稳定的数据集上同样能够获得更高的聚类质量.此外,拒真误差窗口的聚类质量在多数情况下要略高于纳伪误差窗口的聚类质量.然而,在实际应用中应该采用纳伪还是拒真误差窗口,通常还是取决于应用的需求.

CluWin 高质量的聚类结果得益于 EHCF 结构能够准确捕获当前记录的分布状况.旧记录在在线聚类过程中被及时地删除.该淘汰机制使得 EHCF 在簇中心发生偏移时,能保持一个较小的半径,这将使得 CluWin 相对于 CluStream 更加精准,如图 1 所示.尤其是 CluWin 可以保证聚类结果始终是以 ε 为最大相对误差的最近 N 个有效记录构成,从而确保旧记录对簇的影响在在线聚类过程中被及时消除.然而在 CluStream 中,微簇内各记录的时标被累加起来,当且仅当微簇的相对时标低于阈值时,微簇被删除.该策略删除整个微簇而不是单(多)个过期的记录.当簇中心发生漂移时,微簇半径将不断增大,这将混淆不同的簇,并导致聚类质量不够理想.

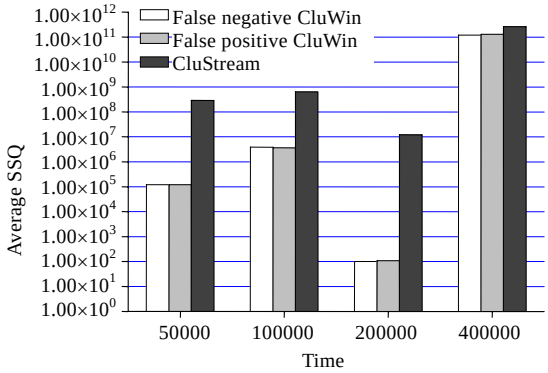


Fig.5 Quality comparison (network intrusion data set, N=10000)

图 5 质量比较(网络入侵检测数据集,N=10000)

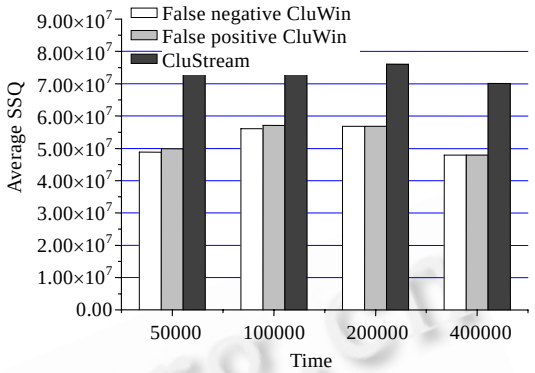


Fig.6 Quality comparison (charitable donation data set, N=4000)

图 6 质量比较(慈善捐赠数据集,N=4000)

5.3 内存开销

数据流处理算法的一个重要特征就是要求算法具有较小的内存空间开销.对于 CluStream 来说,微簇的个数可作为度量单元;而 CluWin 算法中,度量单元可采用 TCF.由于这两种结构的内存代价相近,我们以这两个结构的数目来衡量算法的内存开销.

在 KDD-CUP'99 网络入侵检测数据集中,我们将窗口大小从 10 000 变化到 100 000.由于快照的存储频率会影响到 CluStream 的内存开销,我们设置不同存储频率(从每 1 条记录 1 个快照到每 100 条记录 1 个快照)来进行比较.如图 7 所示,随窗口长度的增加,CluWin 的内存开销恒小于 CluStream,且通常为 2~5 倍的空间节省.

误差参数ε同样会对内存开销造成影响.在网络入侵检测数据集中,CluWin 算法的ε从 0.01 变到 0.1(相应地,CluStream 中各级快照的数目从 100 变到 10).快照存储频率为每 100 条记录 1 个快照.图 8 显示:随着ε的增加,算法内存开销不断地快速降低,且 CluWin 的内存开销仍恒小于 CluStream.

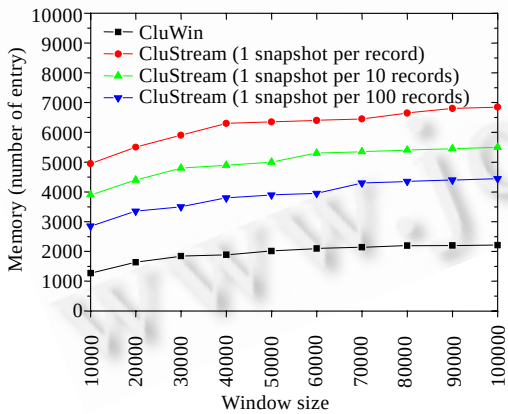


Fig.7 Memory vs. window size (network intrusion data set)

图 7 内存开销随窗口大小变化(网络入侵检测数据集)

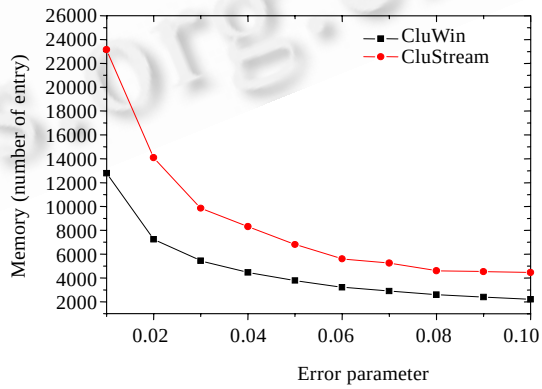


Fig.8 Memory vs. error parameter ε

图 8 内存开销随误差参数ε变化

5.4 聚类处理时间

我们采用真实数据集对 CluWin 和 CluStream 的处理时间进行比较.由于 CluStream 需要定期存储快照,为了加速 CluStream 的处理速度,我们在算法实现时将快照存储在内存中.

首先,我们比较 CluWin 的 EHCF 维护过程和 CluStream 的在线过程的处理时间.在 CluStream 中,快照的存储频率将会影响到处理速度和精度.低的存储频率会提高处理速度,但会造成精度损失.事实上,CluStream 为了获得与 CluWin 相同的精度,需要在每个新记录到达时存储一次快照.然而,这无疑将大大降低 CluStream 的处理速度.为了使算法能够进行比较,我们降低对 CluStream 的精度要求 100 倍,也就是说,每 100 条记录存储 1 个快照.图 9 显示,算法执行时间随数据流长度呈线性增长,且各条直线具有不同的斜率.注意到,低斜率对应高处理速度.因而,CluWin 比 CluStream 要更加高效.

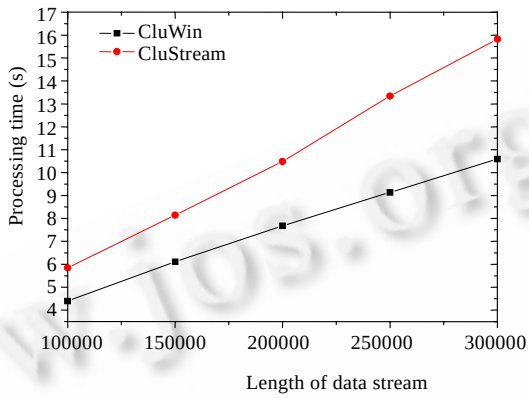


Fig.9 Time vs. length of stream (network intrusion data set)

图 9 处理时间随数据流长度变化(网络入侵检测数据集)

然后,我们采用不同维度和簇个数的数据流对 CluWin 的处理时间进行评价.由于人工数据集可以获得任意的维度和簇个数组合的数据流,我们生成了一系列数据集.第 1 系列数据集的簇个数从 10 变化到 40,并固定数据流的长度和维度.图 10 显示,CluWin 的处理时间随簇的个数呈线性增长.另外一系列数据集的维度从 11 变化到 80,并固定数据流的长度和簇个数.图 11 显示,CluWin 的处理时间随维度线性增长.

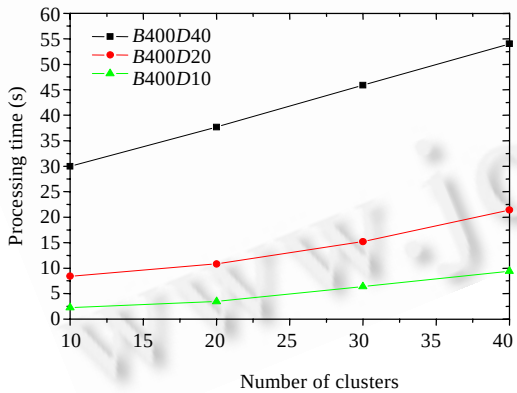


Fig.10 Time vs. number of clusters

图 10 处理时间随簇个数变化

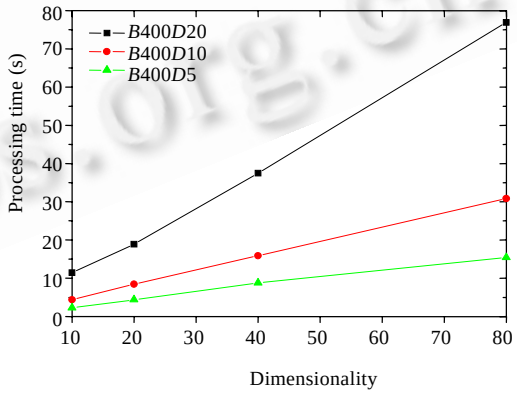


Fig.11 Time vs. dimensionality

图 11 处理时间随维数变化

5.5 参数影响

为了获得较好的聚类质量,EHCF 的个数应当大于自然簇的个数.然而,维护大量的 EHCF 将会导致较大的存储开销和较长的处理时间.为方便下面的讨论,我们引入参数 EHCF 率(EHCF-ratio)用于定义 EHCF 个数与自然簇个数之间的比率.在 KDD-CUP'99 网络入侵数据集中,当前时标 T_c 设置为 100 000;在 KDD-CUP'98 慈善捐赠数据集中,T_c 为 50 000.图 12 显示了 EHCF 率对聚类效果的影响.当 EHCF 率为 1 时,也就是当 EHCF 的个数

与自然簇的个数相等时,聚类质量较差.随着 EHCF 率的增加,平均 SSQ 逐渐减少.当 EHCF 率等于 10 时,平均 SSQ 值趋于稳定.这表明:为了获取较好的聚类质量,EHCF 率并不需要设置过高.

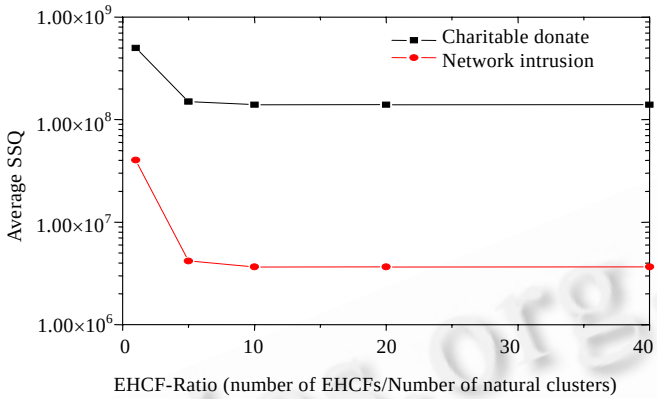


Fig.12 Accuracy impact of EHCF-ratios

图 12 EHCF 率对准确性的影响

6 结论和展望

本文提出一种基于滑动窗口的数据流聚类算法 CluWin.该方法依据实际应用需求,可采用纳伪或拒真两种聚类特征指数直方图作为概要结构,可较好地保存数据流当前窗口内的数据分布状况,从而获取较高质量的聚类结果.理论分析和实验结果表明:本文提出的算法在滑动窗口中相对于现有方法具有更高的聚类质量、更少内存开销和更快的处理速度.今后的工作将对基于数据流聚类的变化检测等相关问题进行研究.

References:

[1] Aggarwal CC, Han J, Wang J, Yu PS. A framework for clustering evolving data streams. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 81−92

[2] Chalaghan LO, Mishra N, Meyerson A, Guha S. Streaming data algorithms for high-quality clustering. In: Proc. of the 18th Int'l Conf. on Data Engineering. San Jose, 2002. 685−694. http://doi.ieeecomputersociety.org/10.1109/ICDE.2002.994785

[3] Domingos P, Hulten C. Mining high-speed data streams. In: Proc. of the KDD. 2000. http://citeseer.ist.psu.edu/domingos00mining.html

[4] Guha S, Meyerson A, Mishra N, Motwani R, Callaghan LO. Clustering data streams: Theory and practice. IEEE Trans. on Knowledge and Data Engineering, 2003,3(15):515−528.

[5] Guha S, Mishra N, Motwani R, Callaghan LO. Clustering data stream. In: Proc. of the 41st Annual Symp. on Foundations of Computer Science. Redondo Beach: IEEE Computer Society, 2000. 359−366.

[6] Nam H, Won S. Statistical grid-based clustering over data streams. SIGMOD Record, 2004,33(1):32−37.

[7] Ordonez C. Clustering binary data streams with *k*-means. In: Zaki MJ, Aggarwal CC, eds. Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD). San Diego, 2003. 12−19.

[8] Zhou A, Cai Z, Wei L, Qian W. M-Kernel merging: Towards density estimation over data streams. In: Proc. of the 8th Int'l Conf. on Database Systems for Advanced Applications (DASFAA). Kyoto, 2003. 285−292.

[9] Aggarwal CC, Han J, Wang J, Yu PS. A framework for projected clustering of high dimensional data streams. In: Nascimento MA, Özsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB, eds. Proc. of the VLDB. Toronto: Morgan Kaufmann Publishers, 2004. 852−863.

[10] Babcock B, Datar M, Motwani R, Callaghan LO. Maintaining variance and *k*-medians over data stream windows. In: Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems. San Diego: ACM Press, 2003. 234−243.

[11] Dai B, Huang J, Yeh M, Chen M. Clustering on demand for multiple data streams. In: Proc. of the ICDM. Brighton, 2004. 367−370.

[12] Nasraoui O, Cardona C, Rojas C, Gonzlez F. TECNO-STREAMS: Tracking evolving clusters in noisy data streams with a scalable immune system learning model. In: Proc. of the 3rd IEEE Int'l Conf. on Data Mining (ICDM). Melbourne, 2003. 235−242.

[13] Yang J. Dynamic clustering of evolving streams with a single pass. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the ICDE. Bangalore, 2003. 695−697.

[14] Han J, Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[15] Charikar M, Callaghan LO, Panigrahy R. Better streaming algorithms for clustering problems. In: Proc. of the 35th Annual ACM Symp. on Theory of Computing. San Diego, 2003. 30−39.

[16] Beringer J, Hullermeier E. Online Clustering of Parallel Data Streams. Data & Knowledge Engineering. 2005. http://wwwiti.cs.uni-magdeburg.de/iti_dke/publications/DKE_draft.pdf

[17] Cao F, Ester M, Qian W, Zhou A. Density-Based clustering over an evolving data stream with noise. In: Proc. of the SIAM Conf. on Data Mining (SDM). 2006.

[18] Zhu WH, Yin J, Xie YH. Arbitrary shape cluster algorithm for clustering data stream. Journal of Software, 2006,17(3):379−387 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/379.htm

[19] Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems. In: Popa L, ed. Proc. of the 21st ACM Symp. on Principles of Databases Systems (PODS). Madison, 2002. 1−16.

[20] Zhang T, Ramakrishnan R, Livny M. Birch: An efficient data clustering method for very large databases. In: Jagadish HV, Mumick IS, eds. Proc. of the SIGMOD. Montreal: ACM Press, 1996. 103−114.

[21] Datar M, Gionis A, Indyk P, Motwani R. Maintaining stream statistics over sliding windows. In: Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA). San Francisco, 2002. 635−644. http://www-cs.stanford.edu/~datar/papers/sicomp_streams.pdf

附中文参考文献:

[18] 朱蔚恒,印鉴,谢益煌.基于数据流的任意形状聚类算法.软件学报,2006,17(3):379−387. http://www.jos.org.cn/1000-9825/17/379.htm

常建龙(1972－),男,博士生,主要研究领域为数据流,数据挖掘.

曹锋(1977－),男,上海崇明人,博士生,主要研究领域为数据流,数据挖掘.

周傲英(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据流,数据挖掘,XML 数据管理,P2P 计算.